

## Numerics

302 HTTP response code 132  
307 HTTP response code 157  
401 HTTP error code 54

## A

access delegation 11–16  
  APIs and 13–14  
  authorization delegation 14  
  overview 11–12  
  user-driven security 15–16  
access tokens. *See also* bearer tokens  
  accepting additional tokens 244  
  authentication and 242–243  
  getting 18  
  overview 63, 133, 141, 287  
  refreshing 313–318  
acr claim 247  
active claim 198  
address attribute 249  
advanced capabilities, OpenID Connect 252–253  
alg header 184  
amr claim 247  
antipattern, password-sharing 102  
API (Application Program Interface) 13–14, 35  
AS (authorization server) 11, 31–32  
  adding refresh token support 86–88  
  adding scope support 88–92  
  authorizing client 77–82

  how server knows client 44–46, 209–210  
  issuing tokens 82–86  
  managing client registrations 76–77  
  protecting bearer tokens 172–173  
  registering clients with 44–46, 209–210  
  vulnerabilities 154–167  
    client impersonation 162–164  
    general security 154–155  
    open redirector 164–167  
    redirect URI manipulation 157–162  
    session hijacking 155–157  
assertion grant types 106–108  
assertions 243  
at\_hash claim 247  
aud claim 185, 247  
audience restriction 244–245  
authentication 236–261  
  common pitfalls 242–246  
  access of protected API as proof of authentication 243–244  
  access tokens as proof of authentication 242–243  
  different protocols for every potential identity provider 245–246  
  injection of access tokens 244  
  injection of invalid user information 245  
  lack of audience restriction 244–245  
  mapping OAuth to 238–241

OpenID Connect 246–261  
  advanced capabilities 252–253  
  calling UserInfo 259–261  
  compatibility with OAuth 2.0 252  
  creating UserInfo endpoint 255–257  
  dynamic server discovery and client registration 250–252  
  generating ID token 254–255  
  ID tokens 246–248  
  parsing ID token 257–259  
  UserInfo endpoint 248–250  
  process of 241–242  
  versus authorization 237–238  
authorization, authentication versus 237–238  
authorization codes  
  acquiring tokens with 46–51  
  adding cross-site protection with state parameter 51  
  processing authorization response 49–50  
  sending authorization request 47–48  
  authorization codes, Proof Key for Code Exchange (PKCE)  
  overview 22–23  
  stealing 134  
  through referrer 128–130  
authorization code grant type  
  as a canonical grant type 93  
  authenticating the resource owner 25, 80

- authorizing the client 26, 79
- choosing a grant type 102
- definition 22
- detail 23
- issuing an access token for an authorization code 29, 82
- issuing an authorization code 82
- native applications 115
- redirecting to the authorization endpoint 24, 47
- redirecting to the client with the authorization code 27, 49, 82
- submitting the authorization code to the token endpoint 28, 49
- authorization delegation 14
- authorization dialog 5
- authorization endpoint 78–79, 315
- authorization framework 5
- authorization grants 22–30, 34–35, 93–109
  - assertion grant types 106–108
  - choosing 108–109
  - client credentials grant type 97–101
  - implicit grant type 94–95
  - processing requests 84–86
  - resource owner credentials grant type 101–106
- Authorization header 28, 29, 30, 52, 60, 61, 199
- authorization-processing mechanisms 19
- authorization protocol 14
- authorization request, sending 47–48, 311
- authorization response, processing 49–50
- authorization server. *See* AS
- authorizationServer.js file 76, 86, 89, 96, 99, 103, 164, 186, 189, 191, 203, 228
- authorized party 247
- auth\_time claim 247
- azp claim 247

## B

- back-channel communication 35–36
- Base64 184
- Basic authentication 83
- Bearer authorization type 292
- Bearer header 53
- Bearer keyword 60, 61
- bearer tokens. *See also* tokens
  - authorization code 173–178
  - defined 168–170
  - overview 29–30, 52
  - protecting 170–173
    - at authorization server 172–173
    - at client 171–172
    - at protected resource 173
  - risks and considerations of 170
- Bearer Token Usage 60
- blacklist trust level 15, 214
- browser applications 110–111

## C

- callbacks, and token request 312
- Cascading Style Sheets. *See* CSS
- CBOR (Concise Binary Object Representation) 196
- certificates 238
- c\_hash claim 247
- claims, JSON Web Token 185–186
- client credentials grant type 97–101
- client\_credentials value 98
- client ID
  - creating 76
  - overview 209
- client\_id\_issued\_at 227
- client\_id parameter 78
- client.js file 76, 89, 175, 217
- client metadata 219–225
  - core client metadata field names 220
  - human-readable values 220–223
  - software statements 223–225
- client\_name 222

- client registration, dynamic server discovery and 250–252
- clients. *See also* dynamic client registration
  - acquiring tokens using authorization code grant type 46–51
    - adding cross-site protection with state parameter 51
  - processing authorization response 49–50
  - sending authorization request 47–48
- at runtime 210–219
  - having client register itself 217–219
  - implementing registration endpoint 214–217
  - protocol 210–212
  - reasons to use dynamic registration 212–213
- authenticating 83–84
- authorizing 77–82
- browser applications 110–111
- client impersonation 162–164
- client metadata 219–225
  - core client metadata field names 220
  - human-readable values 220–223
  - software statements 223–225
- handling secrets 117–118
- how server knows client 209–210
- managing client registrations 76–77
- managing dynamically registered clients 225–235
- native applications 112–117
- overview 5
- protecting bearer tokens 171–172
- refresh access token 54–58
- registering with authorization server 44–46
- using tokens with protected resource 51–54
- vulnerabilities 121–137
  - CSRF attack 122–125

- general client security 121–122
- native applications best practices 136–137
- registration of redirect URI 127–134
- theft of authorization codes 134
- theft of client credentials 125–127
- theft of tokens 134–135
- web applications 109–110
- client secret 28, 44, 114, 117, 125, 126, 162, 227
- client\_uri 222
- CoAP (Constrained Application Protocol) 18, 78
- code, extended code listings 311
- code\_challenge\_method. *See also* PKCE 174, 175, 176
- code framework 305–310
- code parameter 28, 29, 49
- code reuse 20
- code\_verifier. *See also* PKCE 174, 175, 176
- communication
  - back-channel 35–36
  - front-channel 36–39
- “completed” directory 310
- components
  - access tokens 32
  - authorization grants 22–30, 34–35
  - refresh tokens 33–34
  - scopes 32–33
  - tokens 21–22
- Concise Binary Object Representation. *See* CBOR
- confidential clients 44, 117–118
- configuration time secrets 117
- Constrained Application Protocol. *See* CoAP
- consumer key. *See* client secret 117
- contacts 222
- Content Security Policy. *See* CSP
- Content-Type 143, 144
- core client metadata field names 220
- CORS (cross-origin resource sharing) 97, 149

- COSE (CBOR Object Signing and Encryption) 196
- Covert Redirect 161
- credentials
  - sharing 7–11
  - stealing 125–127
- cross-origin resource sharing. *See* CORS
- cross-site protection, adding with state parameter 51
- cross-site scripting. *See* XSS
- cryptographic certificates 26
- cryptographic hash 247
- cryptographic methods 19
- cryptographic protection of JSON Web Tokens (JWT) 188–195
  - asymmetric signatures using RS256 191–195
  - symmetric signatures using HS256 189–191
- cryptography 195
- CSP (Content Security Policy) 147
- CSRF (cross-site request forgery) attack 122–125
- CSS (Cascading Style Sheets) 307

## D

- data results
  - different scopes for 68–70
  - different users for 70–73
- delegating access 11–16
  - APIs and 13–14
  - authorization delegation 14
  - user-driven security 15–16
- delegation protocol 3, 14, 20, 78
- developer key 10, 13
- dynamic client registration 208–235
  - at runtime 210–219
    - having client register itself 217–219
  - implementing registration endpoint 214–217
  - protocol 210–212
  - reasons to use dynamic registration 212–213
- client metadata 219–225
  - core client metadata field names 220

- human-readable values 220–223
- software statements 223–225
- how server knows client 209–210
- managing dynamically registered clients 225–235
  - implementing dynamic client registration management API 228–235
- management protocol 225–228
- Dynamic Client Registration protocol 210
- dynamic server discovery, client registration and 250–252

## E

- elliptical curve cryptography 195
- email attribute 249
- encrypted connections 102
- endpoints
  - authorization endpoint 78–79, 315
  - dynamic client registration 214–217
  - token endpoint 82–86, 316–318
  - token introspection 198–200
  - token revocation 203–204
- UserInfo
  - calling 259–261
  - creating 255–257
  - overview 248–250
- ES256 method 195
- ES384 method 195
- ES512 method 195
- exact matching of redirect URIs 158
- exp claim 185, 247
- expiration timestamp 185
- expires\_in parameter 55, 86–87, 187
- Express.js 60

## F

- federated single-sign-on 26
- FHIR (Fast Healthcare Interoperable Resources) 280

files directory in exercises 307  
 Firebug plugin 25  
 front-channel communication  
   36–39

## G

general security  
   authorization server 154–155  
   client security 121–122  
 Generic Security Services  
   Application Program  
   Interface. *See* GSS-API  
 get-a-token step 169  
 getClient function 198  
   assertion grant types 106–108  
   choosing 108–109  
   client credentials grant type  
     97–101  
   implicit grant type 94–95  
   resource owner credentials  
     grant type 101–106  
 grant\_type parameter 56, 84,  
   98, 103, 107  
 graylist trust level 16, 214  
 GSS-API (Generic Security  
   Services Application  
   Program Interface) 35

## H

header.payload.signature. *See*  
   also: JOSE, JWT 190  
 HEART (Health Relationship  
   Trust) 277–280  
   importance of 277–278  
   mechanical profiles 278–279  
   semantic profiles 280  
   specifications 278  
 HMAC signature algorithm 190  
 HS256 signature method  
   asymmetric signatures  
     191–195  
   symmetric signatures  
     189–191  
 HS384 method 195  
 HS512 method 195  
 HSTS (HTTP Strict Transport  
   Security) 151  
 HTTP 302 redirect 160  
 HTTP 307 redirect 157  
 HTTP Authorization header 60  
 HTTP Basic authentication  
   13, 83

HTTP Digest Auth 13  
 HTTPD Reverse Proxy 295  
 HTTP forms 90  
 HTTP POST messages 203  
 HTTP request, parsing token  
   from 60–62  
 HTTP Strict Transport Security.  
   *See* HSTS  
 HTTP transaction 25  
 human-readable values (client  
   metadata) 220–223

## I

iat claim 185, 247  
 IdP (identity provider)  
   authentication and 245–246  
   overview 239, 279  
 ID tokens  
   generating 254–255  
   overview 246–248  
   parsing 257–259  
 iframe (inline frame) 95  
 iGov (International Government  
   Assurance) 280–281  
   future of 281  
   importance of 280–281  
 IMAP (Internet Message Access  
   Protocol) 213  
 impersonating user 8  
 implicit grant support, for  
   resource endpoints  
     148–151  
 implicit grant type 94, 94–97,  
   95  
 inline frame. *See* iframe  
 inspection tools, Browser 25  
 introspection 196–201  
   building introspection  
     endpoint 198–200  
   combining with JSON Web  
     Token 201  
   introspecting token 200–201,  
     292–294  
   introspection protocol  
     196–198  
 introspection protocol 196–198  
 invalid\_client\_metadata error  
   214  
 invalid user information 245  
 iss claim 185, 247  
 issued-at timestamp 185  
 issuing PoP tokens 284–287,  
   289–291

## J

JOSE (JSON Object Signing  
   and Encryption) 19,  
   188–195, 246, 283  
   additional token  
     protection 195  
   asymmetric signatures using  
     RS256 191–195  
   symmetric signatures using  
     HS256 189–191  
 jti claim name 185  
 JWK (JSON Web Key) 258, 283  
 jwks\_uri 222  
 JWT (JSON Web Token) 19,  
   86, 106, 183–188, 275  
   claims 185–196  
   combining token introspection  
     and 201  
   implementing in servers  
     186–188  
   structure of 183–184

## K

kid claim, Key ID 192  
 keys, PoP tokens 289–291

## L

LDAP (Lightweight Directory  
   Access Protocol)  
   authentication 9  
 lifecycle, of tokens 207  
 localhost 22, 43  
 location.hash 133  
 logo\_uri 222

## M

mechanical profiles, of HEART  
   278–279

## N

native applications 112–117  
 nbf claim name 185  
 node\_modules directory in  
   exercises 306  
 Node Package Manager. *See* NPM  
 nonbearer tokens 30  
 nonce claim 247  
 non-HTTP channels 35  
 nosniff option 146

not-before timestamp 185  
 npm install command 54, 306  
 NPM (Node Package Manager)  
 113, 306

## O

OAuth 2.0 3–20, 298  
 advantages and disadvantages  
 16–18  
 community 302  
 credential sharing 7–11  
 decision-making and  
 299–300  
 defined 3–6  
 delegating access 11–16  
 APIs and 13–14  
 authorization delegation 14  
 user-driven security 15–16  
 ecosystem of 301  
 future of 303–304  
 knowing when to use  
 298–299  
 limitations of 18–20  
 OAuth Authorization Server  
 application. *See also* AS  
 307  
 OAuth Bearer token. *See also*  
 bearer token 29  
 OAuth Client application. *See*  
*also client* 307  
 OAuth Dynamic Client  
 Registration protocol.  
*See also* dynamic client  
 registration 210, 252  
 OAuth Protected Resource  
 application. *See also*  
 protected resources 307  
 OAuth Token Revocation  
 specification. *See also*  
 token revocation 202  
 OCSP (online certificate status  
 protocol) 201  
 OpenID Connect 24, 39, 80,  
 246–261  
 advanced capabilities  
 252–253  
 calling UserInfo 259–261  
 compatibility with OAuth 2.0  
 252  
 creating UserInfo endpoint  
 255–257  
 dynamic server discovery  
 and client registration  
 250–252

ID tokens  
 generating 254–255  
 overview 246–248  
 parsing 257–259  
 UserInfo endpoint  
 calling 259–261  
 creating 255–257  
 overview 248–250  
 openid scope 254, 256  
 open redirector  
 authorization server acting  
 as 164–167  
 stealing tokens through  
 132–134  
 OWASP (Open Web Application  
 Security Project) 122

## P

parsing tokens, from HTTP  
 request 60–62  
 parsing HTTP header 292–294  
 permissions object 275  
 phone attribute 249  
 PKCE (Proof Key for Code  
 Exchange) 109, 118,  
 137, 174–178, 301  
 PKI (public key infrastructure)  
 201, 238  
 policy\_uri 222  
 PoP (Proof of Possession) tokens  
 implementing token support  
 289–294  
 creating signed header and  
 sending to protected  
 resource 291–292  
 introspecting token  
 292–294  
 issuing token and keys  
 289–291  
 parsing header 292–294  
 validating signature  
 292–294  
 requesting and issuing  
 284–287  
 using at protected  
 resource 287–288  
 validating request 288–289  
 private key 194  
 profile attribute 249  
 profiles. *See* protocols and profiles  
 Proof Key for Code Exchange.  
*See* PKCE  
 profile scope 172  
 Proof of Possession. *See* PoP

protected API, authentication  
 and 243–248  
 protected resources 31–32,  
 312  
 creating and sending signed  
 PoP header to 291–292  
 parsing tokens from HTTP  
 request 60–62  
 protecting bearer tokens  
 173  
 serving content based on  
 token 65–74  
 different scopes for different  
 actions 66–68  
 different scopes for different  
 data results 68–70  
 different users for different  
 data results 70–73  
 resource servers 73–74  
 using PoP tokens at 287–288  
 validating tokens against data  
 store 62–65  
 vulnerabilities 138–153  
 endpoints 139–151  
 overview 138–139  
 token replays 151–153  
 protectedResource.js file 86,  
 89, 140, 149, 187, 189  
 protection API, UMA 268  
 protocols and profiles 262–281  
 Health Relationship Trust  
 277–280  
 importance of 277–278  
 mechanical profiles  
 278–279  
 semantic profiles 280  
 specifications 278  
 International Government  
 Assurance 280–281  
 future of 281  
 importance of 280–281  
 User Managed Access  
 263–276  
 importance of 263–265  
 overview 265–276  
 PS256 method 195  
 PS384 method 195  
 PS512 method 195  
 public clients 117–118  
 public key 194

## Q

query parameters 47, 49, 134  
 query strings 90

**R**

redirect, authorization code theft from 157

redirect URI  
manipulation of 157–162  
stealing authorization code through  
referrer 128–137  
stealing token through an open  
redirector 132–134

redirect\_uri parameter 158, 162

Referer header 130, 135, 160

referrers, stealing authorization codes through 128–130

refreshing access tokens 313, 318

refresh tokens  
adding support 86–88  
overview 33–34, 243

registration 208–235  
at runtime 210–219  
having client register  
itself 217–219  
implementing registration  
endpoint 214–217  
protocol 210–212  
reasons to use dynamic  
registration 212–213  
client metadata 219–225  
core client metadata field  
names 220  
human-readable  
values 220–223  
software statements 223–225  
how server knows  
client 209–210  
managing dynamically  
registered clients  
225–235  
implementing dynamic client  
registration management  
API 228–235  
management  
protocol 225–228

registration\_access\_token  
field 225, 227, 231

registration\_client\_uri  
field 225, 227, 231

registration endpoint 126, 320

registration of redirect  
URI 127–134  
stealing authorization  
code through  
referrer 128–130

stealing token through open  
redirector 132–134

relying party. *See* RP

requesting PoP tokens  
284–287

requests variable 79

resource endpoints 139–151  
adding implicit grant  
support 148–151  
protecting 140–148

Resource Owners 22, 31–32

resource owner credentials  
grant type 101–106

resource servers 73–74

response\_types 81, 95, 96, 111,  
115, 211, 212, 218, 222,  
226

revocation 202–207  
implementing revocation  
endpoint 203–204  
revocation protocol 202–203  
revoking token 204–207

RP (relying party) 239

RS384 method 195

RS512 method 195

RSA cryptography 195

RSA key 193

runtime, dynamic client  
registration at 210–219  
having client register  
itself 217–219  
implementing registration  
endpoint 214–217  
protocol 210–212  
reasons to use dynamic  
registration 212–213

runtime secrets 117

**S**

Same Origin Policy 148

SAML (Security Assertion  
Markup Language) 19,  
106

SASL-GSSAPI extension 213

SASL (Simple Authentication  
and Security Layer) 18

SCIM (System for Cross-  
domain Identity  
Management) 212

scopes  
adding scope support 88–92  
overview 32–33, 222

scope parameter 87, 98, 104

secrets (client) 117–118

Secure Sockets Layer. *See* SSL

security  
authorization server 154–167  
acting as open  
redirector 164–167  
client impersonation  
162–164  
general security 154–155  
redirect URI manipulation  
157–162  
session hijacking 155–157  
CSRF attack 122–125  
general client security 121  
JSON Object Signing and  
Encryption 188–195  
additional token protection  
195  
asymmetric signatures using  
RS256 191–195  
symmetric signatures using  
HS256 189–191  
protected resource 138–153  
overview 138  
resource endpoints 139–151  
token replays 151–153  
registration of redirect  
URI 127–134  
stealing authorization code  
through referrer  
128–130  
stealing token through an  
open redirector 132–134  
theft of authorization codes  
134  
theft of client credentials  
125–127  
theft of tokens 134–135

Security Assertion Markup  
Language. *See* SAML

security tokens 26

semantic profiles 280

service-specific password 11, 12

serving content, based on token  
65–74  
different scopes for different  
actions 66–68  
different scopes for different  
data results 68–70  
different users for different  
data results 70–73  
resource servers 73–74

session hijacking, authorization  
server 155–157

shared database 63

signature methods 195



signed header, creating and sending to protected resource 291–292

signed JWT 252

Simple Authentication and Security Layer. *See* SASL

software\_id 222

software statements (client metadata) 223–225

software\_version 222

special passwords, assigning 10

SSL (Secure Sockets Layer) 171

stateless JWT 201

state parameter, adding cross-site protection with 51

stolen token 34

structured tokens, JSON Web Token 183–188

- claims 185–186
- implementing in servers 186–188
- structure of 183–184

sub claim 185, 247

sub field 245

System for Cross-domain Identity Management. *See* SCIM

## T

template object 218

third parties 109

Threat Model Document 18

TLS token binding 294–297

TLS (Transport Layer Security) 151, 154, 171, 283

TOFU (Trust On First Use) 15, 95, 130, 160, 242

token\_endpoint\_auth\_method 212, 219, 221

token endpoint 82–86, 316–318

Token Introspection protocol 19

token revocation endpoint 319

Token Revocation specification 202

tokens 85, 181–207, 282–287

- access 32
- acquiring with authorization code grant type 46–51
- adding cross-site protection with state parameter 51
- processing authorization response 49–50
- sending authorization request 47–48

adding refresh token support 86–88

authorization code 173–178

cryptographic protection of JSON Object Signing and Encryption 188–195

defined 168–170, 181–183

ID tokens

- generating 254–255
- overview 246–248
- parsing 257–259

introspection 196–201

- building introspection endpoint 198–200
- combining introspection and JSON Web Token 201

introspecting token 200–201

introspection protocol 196–198

issuing 82–86

- authenticating client 83–84
- processing authorization grant request 84–86

lifecycle 207

moving beyond bearer tokens 283

overview 21–22

parsing from HTTP request 60–62

PoP tokens 283–289

- implementing token support 289–294
- requesting and issuing 284–287
- using at protected resource 287–288
- validating request 288–289

protecting 170–173

- at authorization server 172–173
- at client 171–172
- at protected resource 173

refresh 33–34

revocation 202–207

- implementing revocation endpoint 203–204
- revoking token 204–207

token revocation protocol 202–203

risks and considerations of 170

serving content based on 65–74

- different scopes for different actions 66–68

- different scopes for different data results 68–70
- different users for different data results 70–73
- resource servers 73–74
- stealing 134–135
- structured 183–188
- TLS token binding 294–297
- validating against data store 62–65

token\_type parameter 29, 86, 87, 204, 287

tos\_uri 222

Transport Layer Security. *See* TLS

trust levels 16

Trust On First Use. *See* TOFU

typ header 184

## U

uma\_authorization 272

uma\_protection 268

UMA (User Managed Access) 263–276

- importance of 263–265
- overview 265–276

unique identifier 185

universal developer key 10

URI (Uniform Resource Identifier) 127–134

- manipulation of 157–162
- stealing authorization code through referrer 128–130
- stealing token through an open redirector 132–134

use-a-token step 169

user approval 316

user authentication 236–261

- authentication versus authorization 237–238
- common pitfalls 242–246
- access of protected API as proof of authentication 243–244
- access tokens as proof of authentication 242–243
- different protocols for every potential identity provider 245–246
- injection of access tokens 244
- injection of invalid user information 245
- lack of audience restriction 244–245

mapping OAuth to 238–241  
OpenID Connect 246–261  
    advanced  
        capabilities 252–253  
    calling UserInfo 259–261  
    compatibility with OAuth 2.0 252  
    creating UserInfo endpoint 255–257  
    dynamic server discovery  
        and client registration 250–252  
    generating ID token 254–255  
    ID tokens 246–248  
    parsing ID token 257–259  
    UserInfo  
        endpoint 248–250  
process of 241–242

user-driven security 15–16  
user\_id field 245  
UserInfo endpoint  
    calling 259–261  
    creating 255–257  
    overview 248–250  
User Managed Access. *See* UMA  
username/password pair 26  
users, impersonating 8  
user's credentials 7, 8, 9, 10  
user-to-user delegation 19

---

**V**

validating request, PoP tokens 288–289  
validating signature, PoP tokens 292–294

validating token against data store 62–65  
validation strategies 132

---

**W**

web applications 38, 109–110  
web client 44  
WebFinger 250, 251  
web server 77  
whitelist 15, 213  
wildcards 158  
WWW-Authenticate header 67

---

**X**

XSS (cross-site scripting) 138  
X-XSS-Protection 146



# OAuth 2 IN ACTION

Richer • Sanso



**T**hink of OAuth 2 as the web version of a valet key. It is an HTTP-based security protocol that allows users of a service to enable applications to use that service on their behalf without handing over full control. And OAuth is used everywhere, from Facebook and Google, to startups and cloud services.

**OAuth 2 in Action** teaches you practical use and deployment of OAuth 2 from the perspectives of a client, an authorization server, and a resource server. You'll begin with an overview of OAuth and its components and interactions. Next, you'll get hands-on and build an OAuth client, an authorization server, and a protected resource. Then you'll dig into tokens, dynamic client registration, and more advanced topics. By the end, you'll be able to confidently and securely build and deploy OAuth on both the client and server sides.

## What's Inside

- Covers OAuth 2 protocol and design
- Authorization with OAuth 2
- OpenID Connect and User-Managed Access
- Implementation risks
- JOSE, introspection, revocation, and registration
- Protecting and accessing REST APIs

Readers need basic programming skills and knowledge of HTTP and JSON.

**Justin Richer** is a systems architect and software engineer. **Antonio Sanso** is a security software engineer and a security researcher. Both authors contribute to open standards and open source.

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit  
[www.manning.com/books/oauth-2-in-action](http://www.manning.com/books/oauth-2-in-action)

“Provides pragmatic guidance on what to do ... and what not to do.”

—From the Foreword by  
Ian Glazer, Salesforce

“Unmatched in both scope and depth. Code examples show how protocols work internally.”

—Thomas O'Rourke  
Upstream Innovations

“A thorough treatment of OAuth 2 ... the authors really know this domain.”

—Travis Nelson  
Software Technology Group

“A complex topic made easy.”

—Jorge Bo, 4Finance IT



\$49.99 / Can \$65.99 [INCLUDING eBook]

ISBN-13: 978-1-61729-327-6  
ISBN-10: 1-61729-327-X



9 781617 293276



5 4 9 9 9